Design Faust DSP with a Web-based Graph Editor

Programmable Audio Workshop 2020

Shihong Ren
shihong.ren@univ-st-etienne.fr

Faust as an interpreter

FAUST

Functional
Audio
Stream

- Write DSP in Faust = available on a large variety of platforms
- Everything in Faust is functional audio stream
- Block-diagram algebra (BDA)

Faust as a text-based language

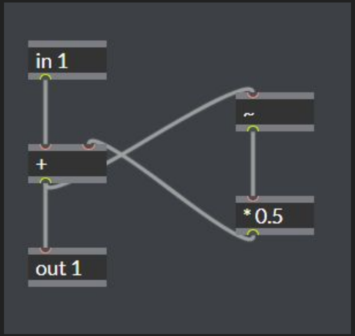
FAUST Functional Audio Stream

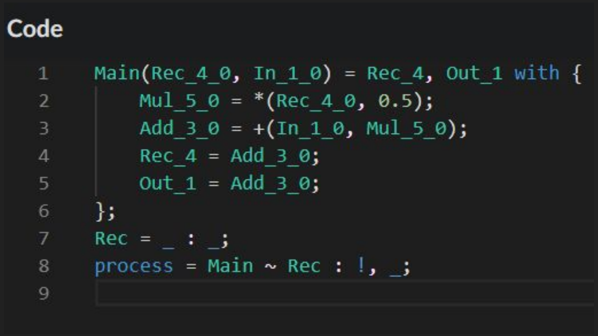
- user-friendliness vs. machine-friendliness
- Block-diagram
- Tools and IDEs

Graph-to-code intepreter

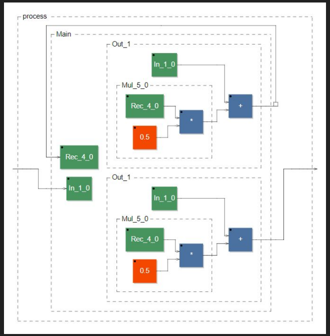
Code-free

- Build an audio graph with boxes and cables
- Generate Faust codes
- Compile Faust codes to a block-diagram / DSP




Code

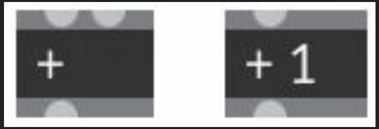
```
Main(Rec_4_0, In_1_0) = Rec_4, Out_1 with {
    Mul_5_0 = *(Rec_4_0, 0.5);
    Add_3_0 = +(In_1_0, Mul_5_0);
    Rec_4 = Add_3_0;
    Out_1 = Add_3_0;
};
Rec = _ : _;
process = Main ~ Rec : !, _;
```

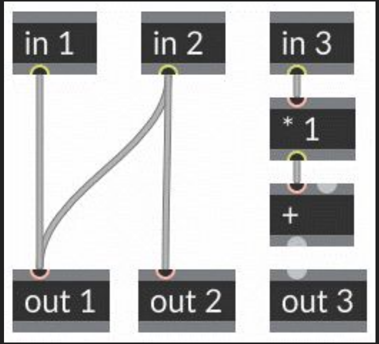


Max/Gen ?

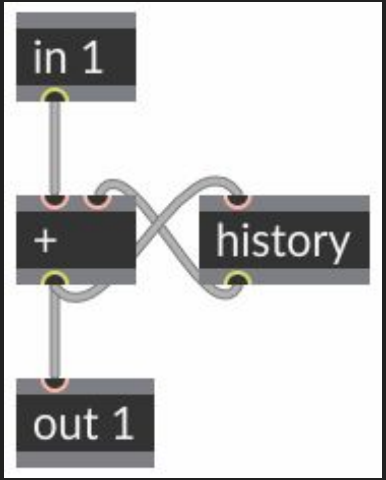
Gen's approach

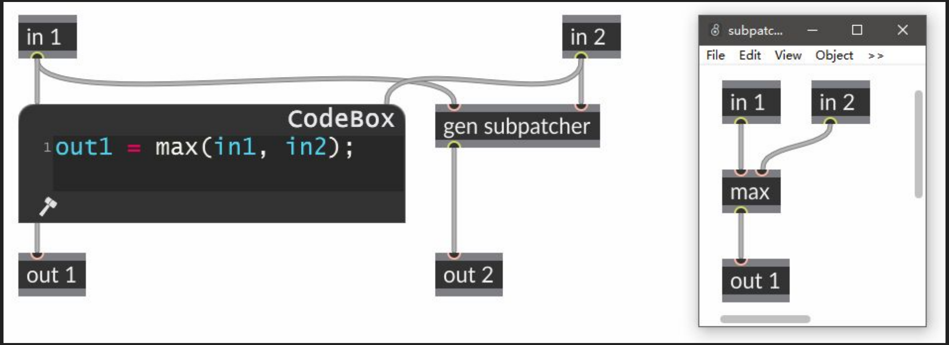
- Operator as box with IO
- Analyze from outputs to inputs
- loops with one-sample delay
- sub-process





```
out1 = in1 + in2;
out2 = in2;
out3 = 0;
```

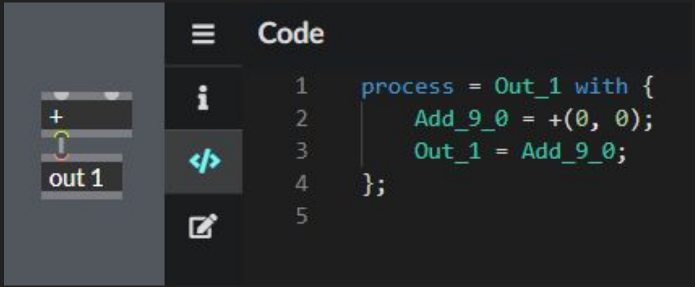




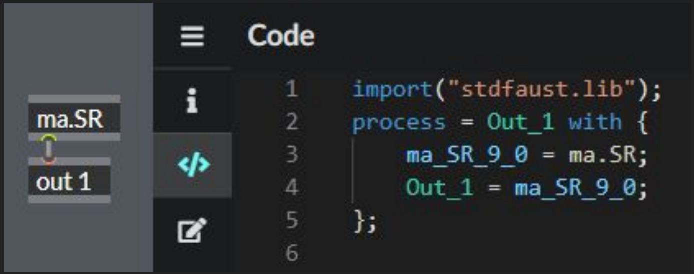
```
out1 = max(in1, in2);
```

How to generate ?

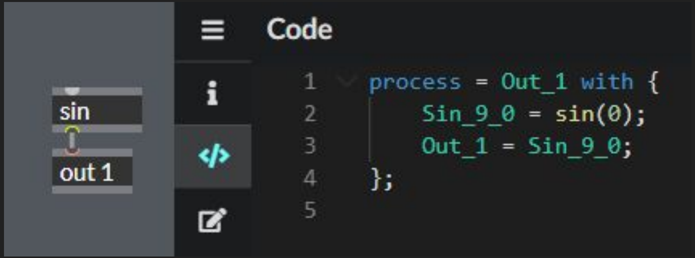
- Functions



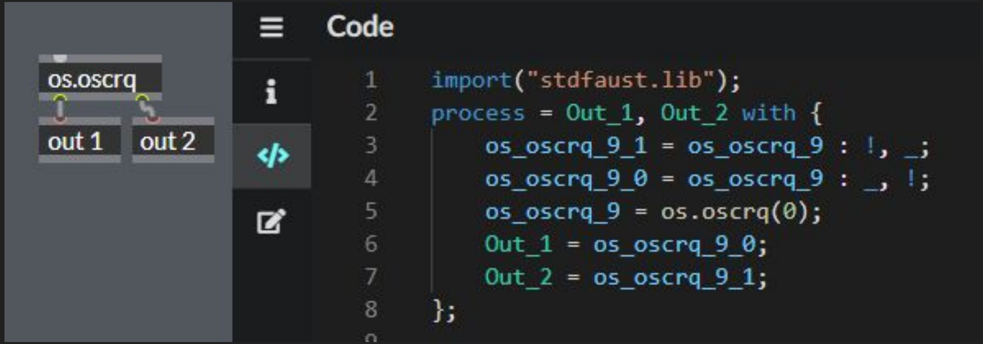
```
process = Out_1 with {
    Add_9_0 = +(0, 0);
    Out_1 = Add_9_0;
};
```



```
import("stdfaust.lib");
process = Out_1 with {
    ma_SR_9_0 = ma.SR;
    Out_1 = ma_SR_9_0;
};
```



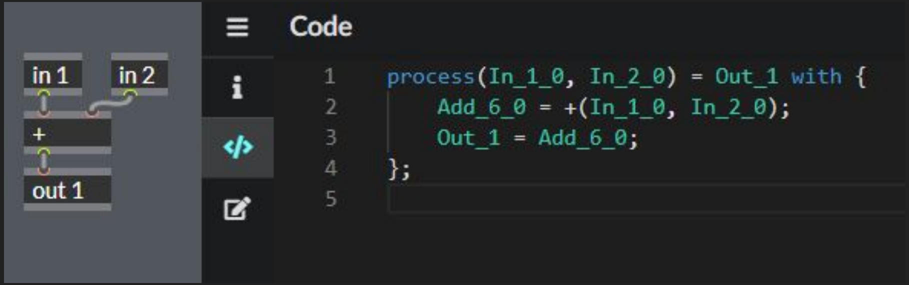
```
process = Out_1 with {
    Sin_9_0 = sin(0);
    Out_1 = Sin_9_0;
};
```



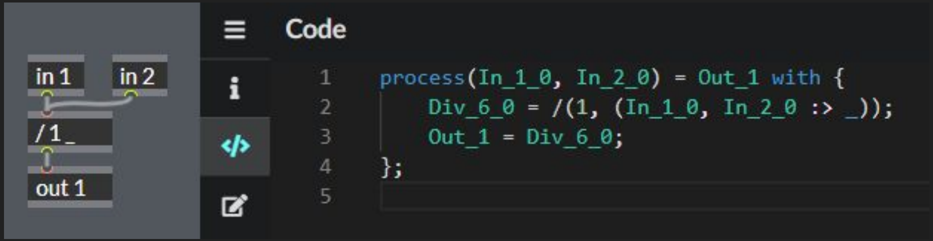
```
import("stdfaust.lib");
process = Out_1, Out_2 with {
    os_oscrq_9_1 = os_oscrq_9 : !, _;
    os_oscrq_9_0 = os_oscrq_9 : _, !;
    os_oscrq_9 = os.oscrq(0);
    Out_1 = os_oscrq_9_0;
    Out_2 = os_oscrq_9_1;
};
```

How to generate ?

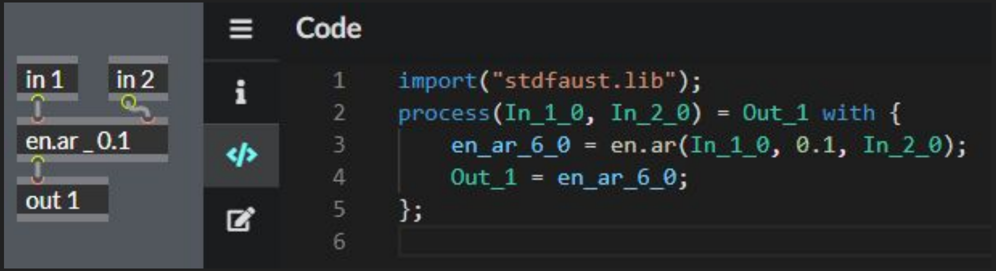
- Functions



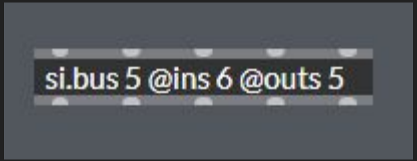
```
process(In_1_0, In_2_0) = Out_1 with {
    Add_6_0 = +(In_1_0, In_2_0);
    Out_1 = Add_6_0;
};
```



```
process(In_1_0, In_2_0) = Out_1 with {
    Div_6_0 = /(1, (In_1_0, In_2_0 :> _));
    Out_1 = Div_6_0;
};
```



```
import("stdfaust.lib");
process(In_1_0, In_2_0) = Out_1 with {
    en_ar_6_0 = en.ar(In_1_0, 0.1, In_2_0);
    Out_1 = en_ar_6_0;
};
```

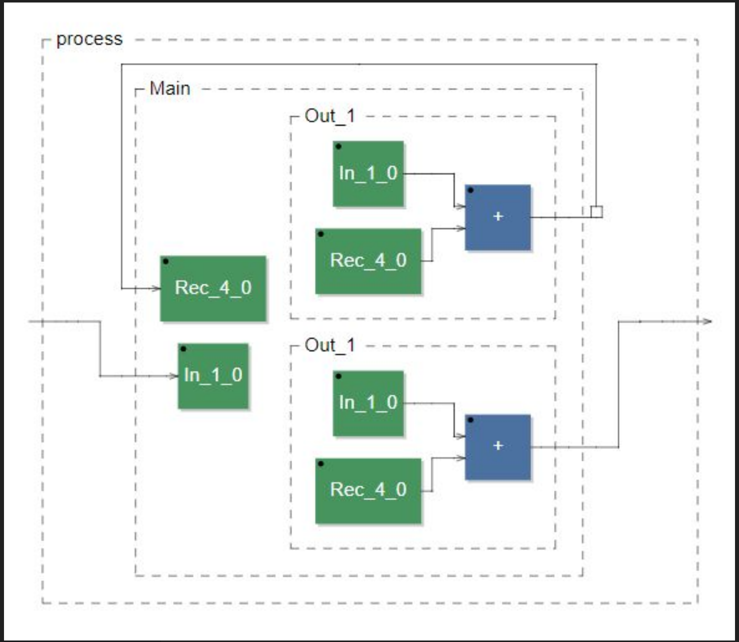


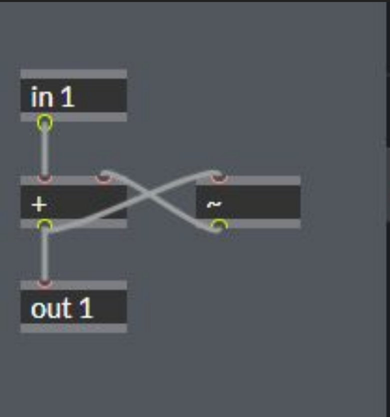
```
si.bus 5 @ins 6 @outs 5
```

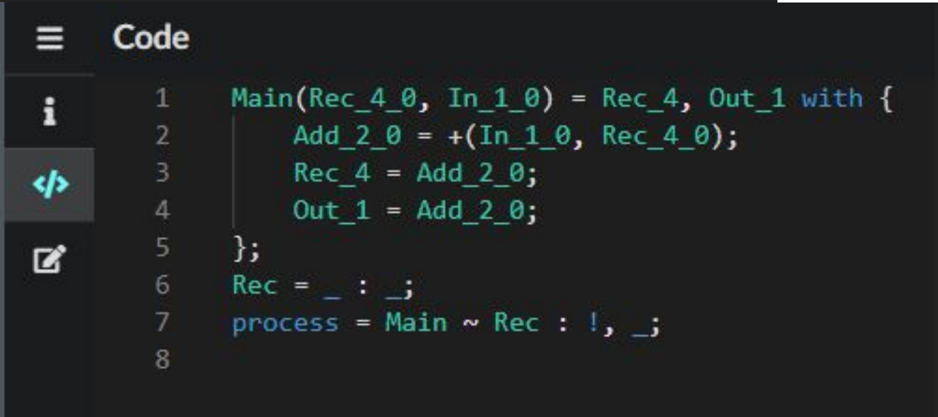
How to generate ?

- Loops

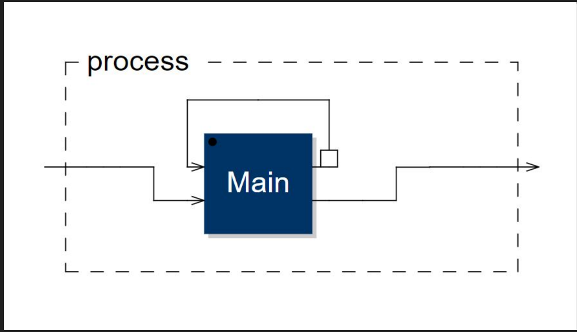
  (recursive composition)





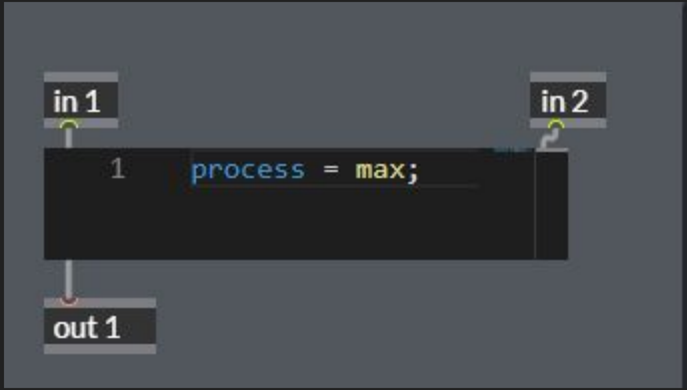


```
Main(Rec_4_0, In_1_0) = Rec_4, Out_1 with {
    Add_2_0 = +(In_1_0, Rec_4_0);
    Rec_4 = Add_2_0;
    Out_1 = Add_2_0;
};
Rec = _ : _;
process = Main ~ Rec : !, _;
```

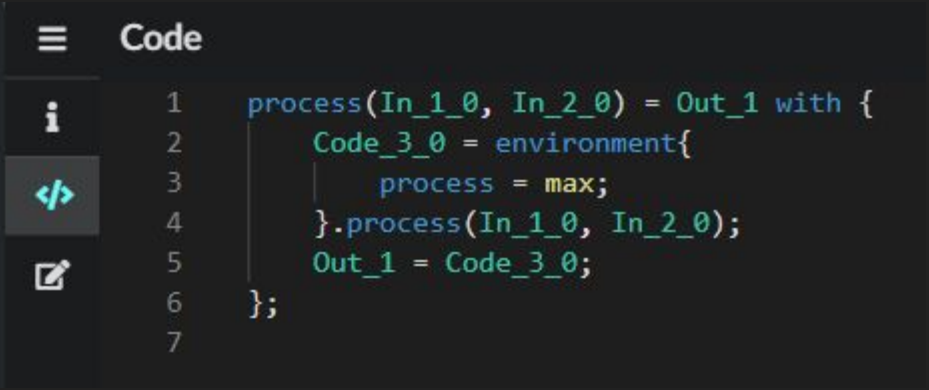


How to generate ?

- subprocess: code-block



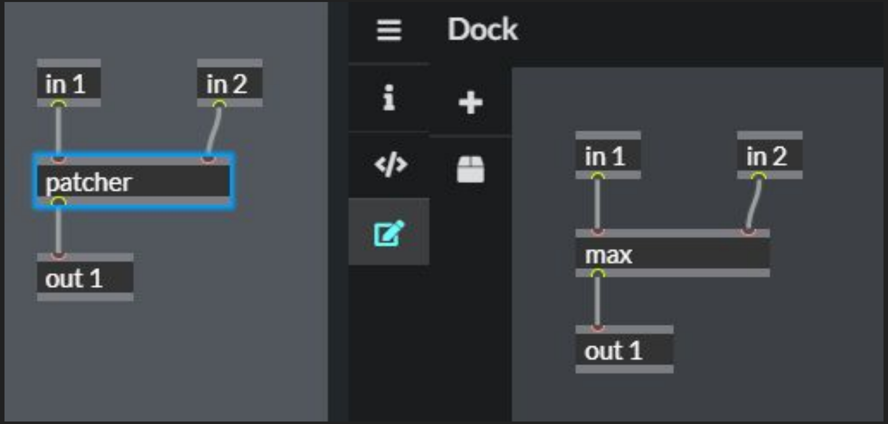
```
process = max;
```

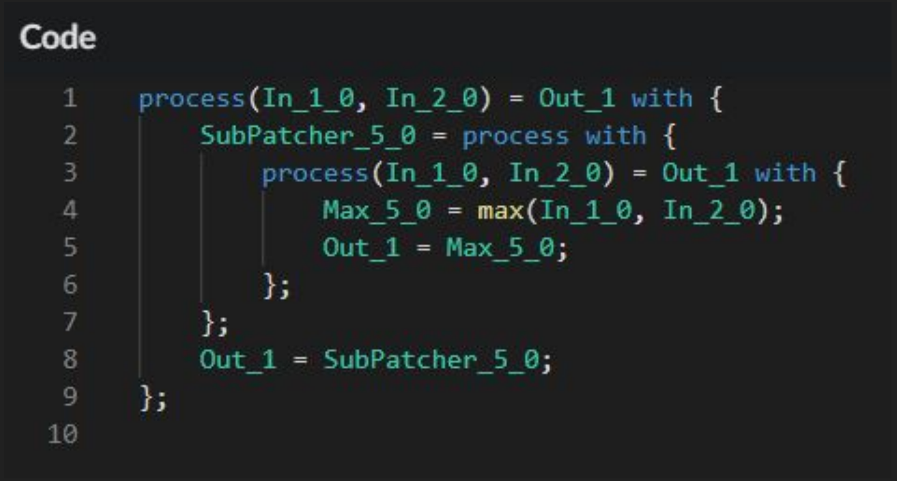


```
process(In_1_0, In_2_0) = Out_1 with {
    Code_3_0 = environment{
        process = max;
    }.process(In_1_0, In_2_0);
    Out_1 = Code_3_0;
};
```

How to generate ?

- subprocess: sub-patcher

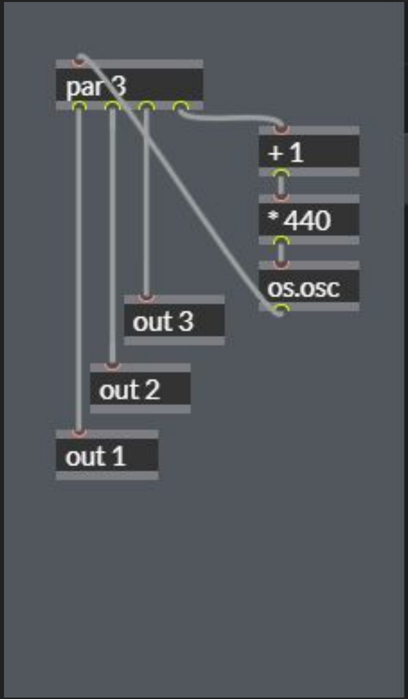


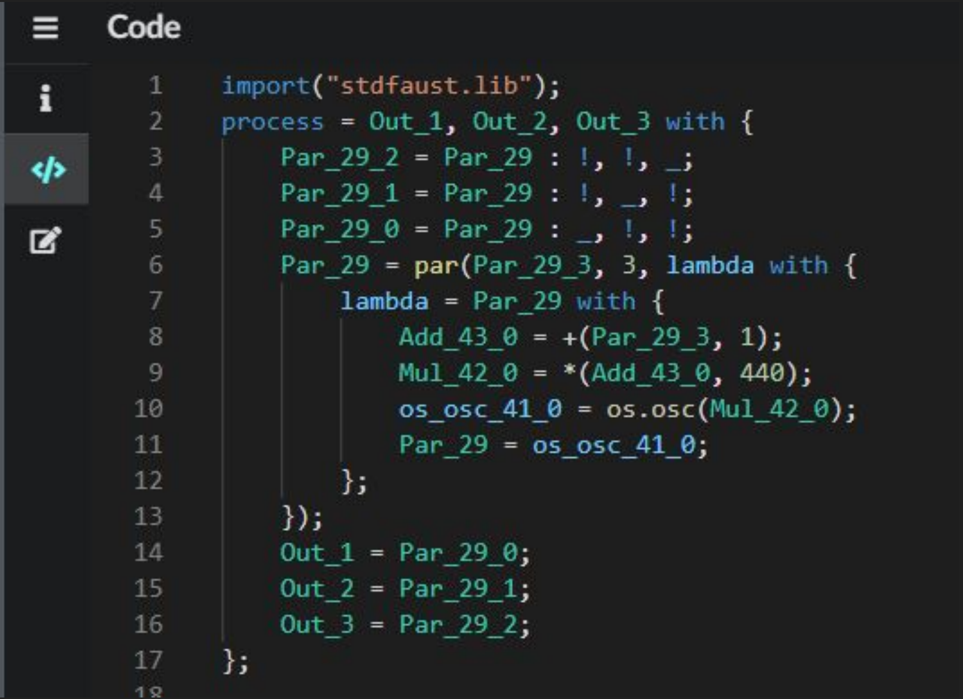

Code

```
process(In_1_0, In_2_0) = Out_1 with {
    SubPatcher_5_0 = process with {
        process(In_1_0, In_2_0) = Out_1 with {
            Max_5_0 = max(In_1_0, In_2_0);
            Out_1 = Max_5_0;
        };
    };
    Out_1 = SubPatcher_5_0;
};
```

How to generate ?

- iterators: par, seq, sum, prod

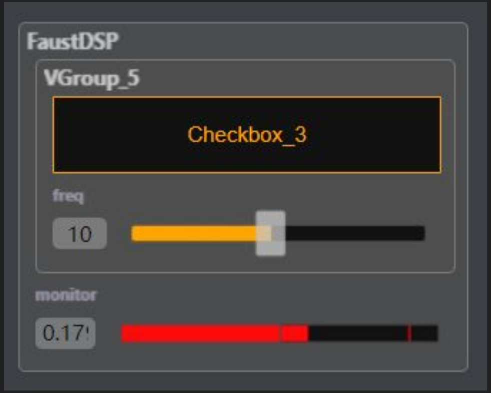


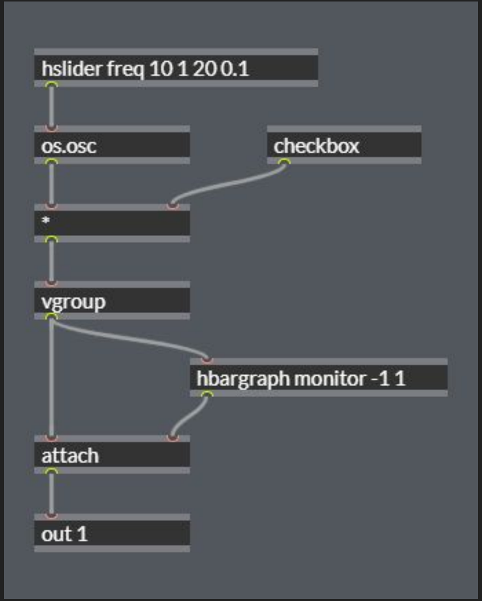


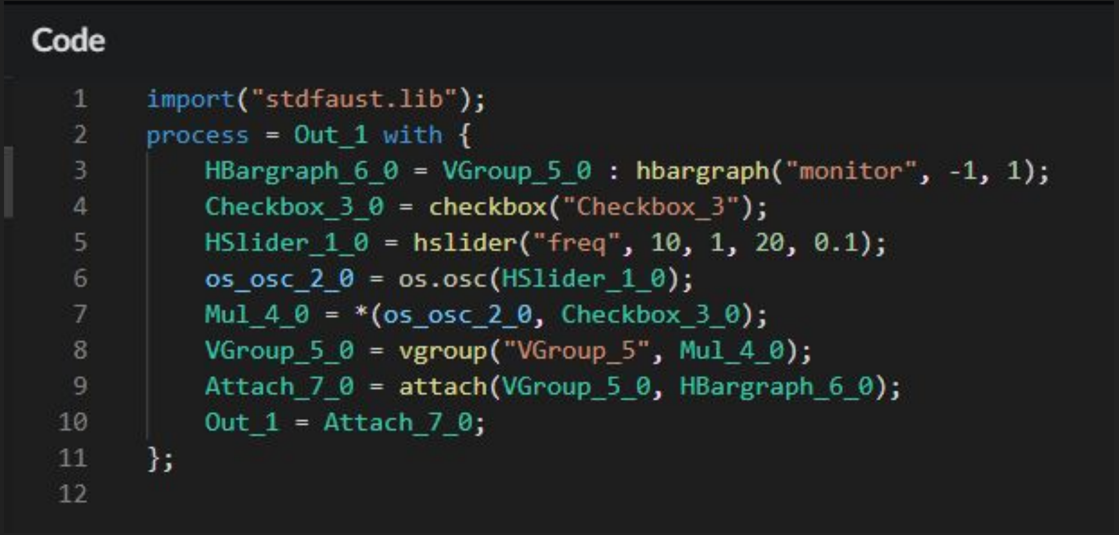
```
import("stdfaust.lib");
process = Out_1, Out_2, Out_3 with {
    Par_29_2 = Par_29 : !, !, _;
    Par_29_1 = Par_29 : !, _, !;
    Par_29_0 = Par_29 : _, !, !;
    Par_29 = par(Par_29_3, 3, lambda with {
        lambda = Par_29 with {
            Add_43_0 = +(Par_29_3, 1);
            Mul_42_0 = *(Add_43_0, 440);
            os_osc_41_0 = os.osc(Mul_42_0);
            Par_29 = os_osc_41_0;
        };
    });
    Out_1 = Par_29_0;
    Out_2 = Par_29_1;
    Out_3 = Par_29_2;
};
```

How to generate ?

- parameters / UI







Code

```
import("stdfaust.lib");
process = Out_1 with {
    HBargraph_6_0 = VGroup_5_0 : hbargraph("monitor", -1, 1);
    Checkbox_3_0 = checkbox("Checkbox_3");
    HSlider_1_0 = hslider("freq", 10, 1, 20, 0.1);
    os_osc_2_0 = os.osc(HSlider_1_0);
    Mul_4_0 = *(os_osc_2_0, Checkbox_3_0);
    VGroup_5_0 = vgroup("VGroup_5", Mul_4_0);
    Attach_7_0 = attach(VGroup_5_0, HBargraph_6_0);
    Out_1 = Attach_7_0;
};
```

Workshop

- JSPatcher (very experimental): https://fr0stbyter.github.io/jspatcher/dist/
- Faust IDE: https://faustide.grame.fr/

Please use the Chrome Browser

Workshop

examples:

- https://fr0stbyter.github.io/jspatcher/dist/?projectZip=../examples/paw.zip
- this link will reset your workspace, make sure you have your project downloaded